樹莓派, 樹莓派之學習, 樹莓派之教育

【鼎革・革鼎】：RASPBIAN STRETCH《六之 J.3・MIR-13.5 □○》

2018-01-11 | 懸鉤子 | 發表迴響

什麼是事物的『特徵』呢？為什麼它的『提取方法』很重要？維基百科詞條這麼說：

Feature extraction

In machine learning, pattern recognition and in image processing, **feature extraction** starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a features vector). This process is called *feature selection*. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

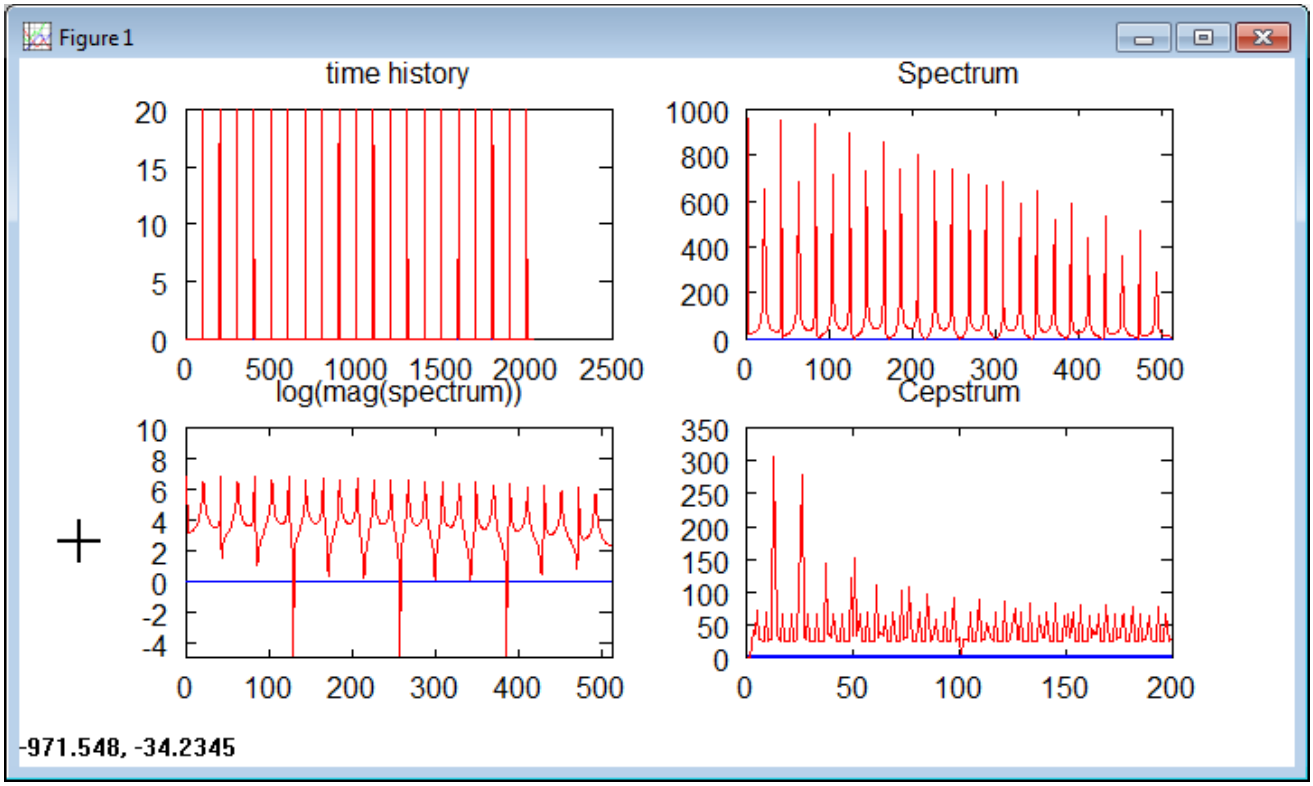
假使考慮如何『定義』事物耶？也許『特徵』就是『界定性徵』，可以用來『區分』相異的東西！所以人們自然懂得『汪星人』不同於『喵星人』的也！！

於是乎好奇那『聲音』本有『調子』，可以用

Cepstrum

A **cepstrum** (/ˈkɛpstrəmˌˈsɛpstrəm/) is the result of taking the Inverse Fourier transform (IFT) of the logarithm of the estimated spectrum of a signal. It may be pronounced in the two ways given, the second having the advantage of avoiding confusion with 'kepstrum' which also exists (see below). There is a *complex cepstrum*, a *real cepstrum*, a *power cepstrum*, and a *phase cepstrum*. The power cepstrum in particular finds applications in the analysis of human speech.

The name "cepstrum" was derived by reversing the first four letters of "spectrum". Operations on cepstra are labelled *quefrency analysis* (aka *quefrency alanysis*[1]), *liftering*, or *cepstral analysis*.



Steps in forming cepstrum from time history

來探討。那麼『圖象』可有『調子』乎？！能否依樣畫葫蘆來研究的呢！？不管『笨鳥先飛』、『菜鳥忘飛』、『老鳥已飛』.....科技史裡滿載『傻問題』之『大成就』矣！！？？何不就效法一下嘛？？！！

——— 《W!o+ 的《小伶鼬工坊演義》：神經網絡【FFT】二》

一門非正式的『特徵工程』

Feature engineering

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive. The need for manual feature engineering can be obviated by automated feature learning.

Feature engineering is an informal topic, but it is considered essential in applied machine learning.

Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering.

— Andrew Ng, Machine Learning and AI via Brain simulations[1]

Features

A feature is an attribute or property shared by all of the independent units on which analysis or prediction is to be done. Any attribute could be a feature, as long as it is useful to the model.

The purpose of a feature, other than being an attribute, would be much easier to understand in the context of a problem. A feature is a characteristic that might help when solving the problem.[2]

Importance of features

The features in your data are important to the predictive models you use and will influence the results you are going to achieve. The quality and quantity of the features will have great influence on whether the model is good or not.[3]

You could say the better the features are, the better the result is. This isn't entirely true, because the results achieved also depend on the model and the data, not just the chosen features. That said, choosing the right features is still very important. Better features can produce simpler and more flexible models, and they often yield better results.[2]

The algorithms we used are very standard for Kagglers. […] We spent most of our efforts in feature engineering. […] We were also very careful to discard features likely to expose us to the risk of over-fitting our model.

— Xavier Conort, "Q&A with Xavier Conort"[4]

…some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.

— Pedro Domingos, "A Few Useful Things to Know about Machine Learning"[5]

道出『創建特徵』並不容易。今日與其說它是『工程』，反倒更類似『藝術』？學法宜先浸醞在『特徵殿堂』也耶！

Feature extraction

Spectral features

`chroma_stft`([y, sr, S, norm, n_fft, …])	Compute a chromagram from a waveform or power spectrogram.
`chroma_cqt`([y, sr, C, hop_length, fmin, …])	Constant-Q chromagram
`chroma_cens`([y, sr, C, hop_length, fmin, …])	Computes the chroma variant "Chroma Energy Normalized" (CENS), following [R15].
`melspectrogram`([y, sr, S, n_fft, …])	Compute a mel-scaled spectrogram.
`mfcc`([y, sr, S, n_mfcc])	Mel-frequency cepstral coefficients

`rmse`([y, S, frame_length, hop_length, …])	Compute root-mean-square (RMS) energy for each frame, either from the audio samples *y* or from a spectrogram *S*.
`spectral_centroid`([y, sr, S, n_fft, …])	Compute the spectral centroid.
`spectral_bandwidth`([y, sr, S, n_fft, …])	Compute p’th-order spectral bandwidth:
`spectral_contrast`([y, sr, S, n_fft, …])	Compute spectral contrast [R16]
`spectral_rolloff`([y, sr, S, n_fft, …])	Compute roll-off frequency
`poly_features`([y, sr, S, n_fft, hop_length, …])	Get coefficients of fitting an nth-order polynomial to the columns of a spectrogram.
`tonnetz`([y, sr, chroma])	Computes the tonal centroid features (tonnetz), following the method of [R17].
`zero_crossing_rate`(y[, frame_length, …])	Compute the zero-crossing rate of an audio time series.

Rhythm features

`tempogram`([y, sr, onset_envelope, …])	Compute the tempogram: local autocorrelation of the onset strength envelope.

Feature manipulation

`delta`(data[, width, order, axis, trim])	Compute delta features: local estimate of the derivative of the input data along the selected axis.
`stack_memory`(data[, n_steps, delay])	Short-term history embedding: vertically concatenate a data vector or matrix with delayed copies of itself.